

# Homework 3

计 52 宋世虹 2015011267

2017 年 6 月

## 1 对象与函数

对象：

- Node: 表示 seam dp 时每一个点的权值和路径中上一个点的值。
- type : 表示当前要 carve 的是列还是行。
- newPoint : 表示 cut 完的 seam 中的一个点。
- T : 双向缩放的时候所用的数据结构, 其中存有: 当前图片, seam 掉一行或一列后的图片, seam 掉一行的路径, seam 掉一列的路径, seam 掉一行的 cost, seam 掉一列的 cost, 上一次 choose 了 seam 掉一行还是一列, 当前的图片的 seam 权值。

函数：

- inImage : 判断一个点是否在图片内。
- calculateEnergy : 计算论文中所述能量, 可以用不同算子。
- DP : 通过 energy 函数计算 seam, dp 出全图每点的权值, 存在 seam 中。
- calculateMin : 根据 seam 图计算最小的 seam, 返回最小的那一列(行)标。
- removeline : 根据已得的 seam 移去一行或一列。
- addLine : 根据已得的 seam 重新在删去 seam 的图中加上一条红色的 seam。
- getInfo : 对于一个已知图片的 T, 计算它的行列 seam。
- onMouse : 将鼠标选中的区域染红。

## 2 seam 的求解

使用上述 calculate energy 和 dp 函数即可。

代码：

```
1 void calculateEnergy(Mat& input, double** energy, int row, int col)
2 {
3     Mat energyOutput;
4     // Mat energyX, energyY, abs_grad_x, abs_grad_y;
5     // Sobel(input, energyX, -1, 1, 0, 3, 1, 0, BORDER_DEFAULT);
6     // Sobel(input, energyY, -1, 0, 1, 3, 1, 0, BORDER_DEFAULT);
7     // convertScaleAbs(energyX, abs_grad_x);
8     // convertScaleAbs(energyY, abs_grad_y);
9     // addWeighted(abs_grad_x, 0.5, abs_grad_y, 0.5, 0, energyOutput);
10    Laplacian(input, energyOutput, -1, 1, 1, 0, BORDER_DEFAULT);
11    for (int i = 0; i < row; ++i)
```

```

12     for ( int j = 0;j < col;++j )
13     {
14         energy [ i ] [ j ] = 0;
15         for ( int k = 0;k < 3;++k )
16             energy [ i ] [ j ] += ( int )(energyOutput .at<Vec3b>(i , j ) [ k ] );
17     }
18 }
19
20 void DP(Node** seam , double** energy ,int row ,int col )
21 {
22     for ( int i = 0;i < col;++i )
23     {
24         seam [ 0 ] [ i ].number = energy [ 0 ] [ i ];
25         seam [ 0 ] [ i ].last = -1;
26     }
27     for ( int i = 1;i < row; ++i )
28         for ( int j = 0;j < col; ++j )
29     {
30         for ( int p = j -1;p <= j +1; ++p )
31             if ( p < 0 || p >= col ) continue ;
32             else
33             {
34                 if ( seam [ i -1] [ p ].number + energy [ i ] [ j ] <= seam [ i ] [ j ].number )
35                 {
36                     seam [ i ] [ j ].number = seam [ i -1] [ p ].number + energy [ i ] [ j ];
37                     seam [ i ] [ j ].last = p ;
38                 }
39             }
40     }
41 }

```

calculatEnergy 用了不同的算子，会在后文中提到，energy 会用这三通道的输出结果的直接和作为真正的 energy。

DP 计算了论文中所给的方法，每次选择上面的左中右三个 pixel 的 seam 权值，来得到当前最小的 seam 权值。在计算的过程中记录 seam 的路径。

### 3 使用不同算子进行测试

笔者使用了 Sobel 算子和拉普拉斯算子。以下是效果：

原图：(之后横纵都 cut 10%)



Laplacian :



Sobel :



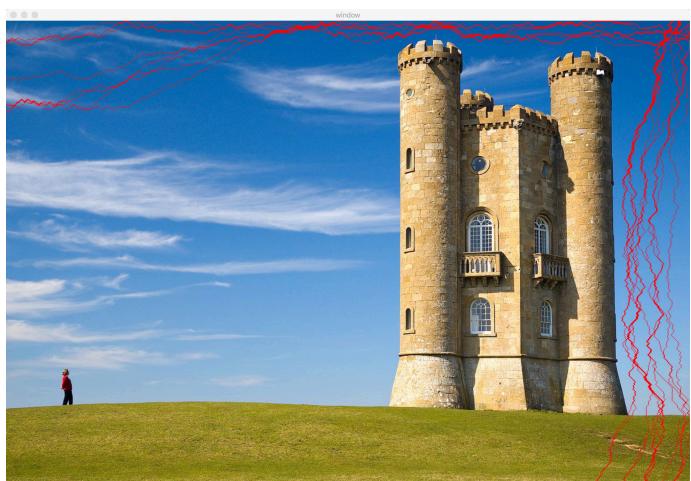
原图 : (之后横纵都 cut 1% )



Laplacian :



Sobel :





对应的代码段：

```

1 void calculateEnergy(Mat& input ,double** energy ,int row,int col)
2 {
3     Mat energyOutput;
4     // Mat energyX ,energyY ,abs_grad_x ,abs_grad_y ;
5     // Sobel( input ,energyX ,-1,1,0,3,1,0,BORDER_DEFAULT) ;
6     // Sobel( input ,energyY ,-1,0,1,3,1,0,BORDER_DEFAULT) ;
7     // convertScaleAbs( energyX , abs_grad_x );
8     // convertScaleAbs( energyY , abs_grad_y );
9     // addWeighted( abs_grad_x , 0.5 , abs_grad_y , 0.5 , 0,energyOutput );
10    Laplacian( input ,energyOutput ,-1,1,1,0,BORDER_DEFAULT) ;
11    for ( int i = 0;i < row;++i )
12        for ( int j = 0;j < col;++j )
13        {
14            energy [ i ][ j ] = 0;
15            for ( int k = 0;k < 3;++k )
16                energy [ i ][ j ] += ( int )(energyOutput .at<Vec3b>(i ,j )[ k ]) ;
17        }
18 }
```

## 4 图像放大、双向缩放

双向缩放包括了图像放大，所以这里只写双向缩放了。

先说双向缩放的缩小。不同于单向缩放，双向缩放要考虑两边的放大和缩小，所以不能直接用直接暴力的 dp 来实现。论文中给出的算法是对于每次 cut 掉 r 行 c 列的图来 dp，新的图可以是被 cut 掉 r-1 行 c 列后又 cut 一行的结果，也可以是被 cut 掉 r 行 c-1 列又 cut 一列的结果。比较这两者之前的 cost 加上 seam 掉那一行或者一列的结果，我们就得到了新的 cost，使用这个 cost 我们选择出了新的图，并计算出行列 carve 之后的结果，用这样的结果再去做新的 dp。下面是代码：

```

1 void getInfo(T& picture ,bool** choosed = NULL)
2 {
3     int row = picture .pic .rows ;
4     int col = picture .pic .cols ;
5     double** energy = new double *[row];
6     for ( int i = 0;i < row;++i )
7         energy [ i ] = new double [ col ];
```

```

8 calculateEnergy(picture.pic, energy, row, col);
9 if (choosed != NULL)
10    for (int i = 0; i < row; ++i)
11       for (int j = 0; j < col; ++j)
12          if (choosed[i][j]) energy[i][j] = -INT_MAX;
13 Node** seam = new Node*[row];
14 for (int i = 0; i < row; ++i)
15    seam[i] = new Node[col];
16 for (int i = 0; i < row; ++i)
17    for (int j = 0; j < col; ++j)
18       seam[i][j].number = INT_MAX;
19 DP(seam, energy, row, col);
20 int removenewPoint = calculateMin(seam, row, col, picture.cseam);
21 type nowType = Col;
22 picture.colpic = removeLine(seam, picture.pic, removenewPoint, row, col, picture.
23     colseam, nowType);
24 picture.pic = picture.pic.t(), nowType = Row;
25 row = picture.pic.rows;
26 col = picture.pic.cols;
27 double** energyr = new double*[row];
28 for (int i = 0; i < row; ++i)
29    energyr[i] = new double[col];
30 calculateEnergy(picture.pic, energyr, row, col);
31 if (choosed != NULL)
32    for (int i = 0; i < col; ++i)
33       for (int j = 0; j < row; ++j)
34          if (choosed[i][j]) energyr[j][i] = -INT_MAX;
35 Node** seamr = new Node*[row];
36 for (int i = 0; i < row; ++i)
37    seamr[i] = new Node[col];
38 for (int i = 0; i < row; ++i)
39    for (int j = 0; j < col; ++j)
40       seamr[i][j].number = INT_MAX;
41 DP(seamr, energyr, row, col);
42 removenewPoint = calculateMin(seamr, row, col, picture.rseam);
43 picture.rowpic = removeLine(seamr, picture.pic, removenewPoint, row, col, picture.
44     rowseam, nowType).t();
45 picture.pic = picture.pic.t();
46 }
47 vector<vector<T>> dpphoto;
48 vector<T> gg(totcol + 1);
49 for (int i = 0; i < totrow + 1; ++i)
50    dpphoto.push_back(gg);
51 dpphoto[0][0].pic = input;
52 dpphoto[0][0].nowgain = 0;
53 getInfo(dpphoto[0][0]);
54 for (int i = 0; i <= totrow; ++i)
55    for (int j = 0; j <= totcol; ++j)
56    {
57       if (i == 0 && j == 0) continue;
58       else if (i == 0)
59       {
60          dpphoto[i][j].pic = dpphoto[i][j - 1].colpic;
61          dpphoto[i][j].nowgain += dpphoto[i][j - 1].cseam;
62          dpphoto[i][j].choose = Col;
63          getInfo(dpphoto[i][j]);
64       }
65       else if (j == 0)
66       {
67          dpphoto[i][j].pic = dpphoto[i - 1][j].rowpic;
68          dpphoto[i][j].nowgain += dpphoto[i - 1][j].rseam;
69          dpphoto[i][j].choose = Row;
70          getInfo(dpphoto[i][j]);
71       }
72    }
73 }
74 }
```

```

69 }
70 else
71 {
72     if (dpphoto[i-1][j].nowgain + dpphoto[i-1][j].rseam > dpphoto[i][j-1].
73         nowgain + dpphoto[i][j-1].cseam)
74     {
75         dpphoto[i][j].nowgain = dpphoto[i][j-1].nowgain + dpphoto[i][j-1].cseam
76         ;
77         dpphoto[i][j].choose = Col;
78         dpphoto[i][j].pic = dpphoto[i][j-1].colpic;
79         getInfo(dpphoto[i][j]);
80     }
81     else
82     {
83         dpphoto[i][j].nowgain = dpphoto[i-1][j].nowgain + dpphoto[i-1][j].rseam
84         ;
85         dpphoto[i][j].choose = Row;
86         dpphoto[i][j].pic = dpphoto[i-1][j].rowpic;
87         getInfo(dpphoto[i][j]);
88     }
89 }
90 Mat seamPic = dpphoto[totrow][totcol].pic;
91 imwrite("result.jpg",dpphoto[totrow][totcol].pic);
92 int r = totrow,c = totcol;
93 while (r >= 0 && c >= 0)
94 {
95     if (r == 0 && c == 0) break;
96     // printf("%d %d\n",r,c);
97     if (dpphoto[r][c].choose == Col)
98     {
99         --c;
100        if (c < 0) break;
101        Mat pic = Mat(seamPic.rows,seamPic.cols+1,CV_8UC3);
102        for (int j = seamPic.rows-1;j >= 0;--j)
103        {
104            // printf("%d\n",j );
105            int num = 0;
106            for (int p = 0;p < seamPic.cols;++p)
107                if (p == dpphoto[r][c].colseam[j].x)
108                {
109                    pic.at<Vec3b>(dpphoto[r][c].colseam[j].y,num++) = Vec3b(0,0,255);
110                    pic.at<Vec3b>(dpphoto[r][c].colseam[j].y,num++) = seamPic.at<Vec3b>(
111                        dpphoto[r][c].colseam[j].y,p);
112                }
113            seamPic = pic;
114        }
115    else
116    {
117        --r;
118        if (r < 0) break;
119        Mat pic = Mat(seamPic.rows+1,seamPic.cols,CV_8UC3);
120        for (int j = seamPic.cols-1;j >= 0;--j)
121        {
122            int num = 0;
123            for (int p = 0;p < seamPic.rows;++p)
124                if (p == dpphoto[r][c].rowseam[j].y)
125                {
126                    pic.at<Vec3b>(num++,dpphoto[r][c].rowseam[j].x) = Vec3b(0,0,255);

```

```

127     pic . at<Vec3b>(num++,dpphoto[ r ][ c ].rowseam[ j ].x) = seamPic . at<Vec3b>(p
128         ,dpphoto[ r ][ c ].rowseam[ j ].x);
129     }
130     else
131         pic . at<Vec3b>(num++,dpphoto[ r ][ c ].rowseam[ j ].x) = seamPic . at<Vec3b>(p
132             ,dpphoto[ r ][ c ].rowseam[ j ].x);
133     }
134     seamPic = pic ;
135 }
136 imshow( "window" ,seamPic );
waitKey(0);

```

getInfo 通过当前图得到 seam 掉一行或一列后的图片, seam 掉一行的路径, seam 掉一列的路径, seam 掉一行的 cost, seam 掉一列的 cost, 上一次 choose 了 seam 掉一行还是一列, 当前的图片的 seam 权值。

然后主函数里面我们就对图片进行 dp, 计算出切割掉 totrow,totcol 行之后的图片, 然后从这张图片再还原 seam。

效果如算子测试中图所示。

双向放大的方法是, 对行列分别放大。因为行列等价, 我们这里只说列的放大方法。对于要放大的列数了 k, 先找出这张图的 k 条 seam, 然后把这 k 条 seam duplicate。对于找 k 条 seam, 我的方法是每次把找到的 seam 的 energy 函数调得很大, 这样就可以直接找到别的 seam 而不找这一条了。这样我们就能找到 k 条 seam 了。

代码如下 :

```

1 int totrow= atof(argv[4])*temp.rows;
2 int totcol = atof(argv[3])*temp.cols;
3 double** energy = new double*[row];
4 for (int i = 0;i < row;++i)
5     energy[ i ] = new double[ col ];
6 unsigned short** addtime = new unsigned short*[row];
7 for (int i = 0;i < row;++i)
8     addtime[ i ] = new unsigned short[ col ];
9 for (int i = 0;i < row;++i)
10    for (int j = 0;j < col;++j)
11        addtime[ i ][ j ] = 0;
12 calculateEnergy(input,energy,row,col);
13 Node** seam = new Node*[row];
14 for (int i = 0;i < row;++i)
15     seam[ i ] = new Node[ col ];
16 double meiyongde = 0;
17 while (totcol--)
{
18     for (int i = 0;i < row;++i)
19         for (int j = 0;j < col;++j)
20             seam[ i ][ j ].number = INT_MAX;
21 DP(seam,energy,row,col);
22 int removenewPoint = calculateMin(seam,row,col,meiyongde);
23 int t = row-1;
24 while (removenewPoint != -1)
{
25     energy[ t ][ removenewPoint ] = INT_MAX;
26     addtime[ t ][ removenewPoint ]++;
27     removenewPoint = seam[ t ][ removenewPoint ].last ;
28     t--;
29 }
30 }
31 // for (int i = 0;i < row++)
32 // delete [] energy[ i ],seam[ i ],addtime[ i ];
33 // delete [] energy,seam,addtime;
34 Mat amplifyPic = Mat(temp.rows,temp.cols + atof(argv[3])*temp.cols,CV_8UC3);
35 for (int i = 0;i < temp.rows;++i)

```

```

38     for (int j = 0,k = 0;j < temp.cols;++j)
39         for (int p = 0;p <= addtime[i][j];++p)
40             amplifyPic.at<Vec3b>(i ,k++) = temp.at<Vec3b>(i ,j );
41     amplifyPic = amplifyPic.t();
42     row = amplifyPic.rows;
43     col = amplifyPic.cols;
44     double** energyr = new double*[row];
45     for (int i = 0;i < row;++i)
46         energyr[i] = new double[col];
47     calculateEnergy(amplifyPic ,energyr ,row ,col);
48     Node** seamr = new Node*[row];
49     for (int i = 0;i < row;++i)
50         seamr[i] = new Node[col];
51     unsigned short** addtimenew = new unsigned short*[row];
52     for (int i = 0;i < row;++i)
53         addtimenew[i] = new unsigned short[col];
54     for (int i = 0;i < row;++i)
55         for (int j = 0;j < col;++j)
56             addtimenew[i][j] = 0;
57     while (totrow--)
58     {
59         for (int i = 0;i < row;++i)
60             for (int j = 0;j < col;++j)
61                 seamr[i][j].number = INT_MAX;
62         DP(seamr ,energyr ,row ,col);
63         int removenewPoint = calculateMin(seamr ,row ,col ,meiyongde);
64         int t = row-1;
65         while (removenewPoint != -1)
66         {
67             energyr[t][removenewPoint] = INT_MAX;
68             addtimenew[t][removenewPoint]++;
69             removenewPoint = seamr[t][removenewPoint].last ;
70             t--;
71         }
72     }
73     Mat amplify = Mat(temp.cols + atof(argv[3])*temp.cols ,temp.rows + atof(argv[4])
74                         *temp.rows,CV_8UC3);
75     for (int i = 0;i < amplifyPic.rows;++i)
76         for (int j = 0,k = 0;j < amplifyPic.cols;++j)
77             if (addtimenew[i][j] == 0)
78                 amplify.at<Vec3b>(i ,k++) = amplifyPic.at<Vec3b>(i ,j );
79             else
80             {
81                 for (int p = 0;p <= addtimenew[i][j];++p)
82                     amplify.at<Vec3b>(i ,k++) = amplifyPic.at<Vec3b>(i ,j );
83             }
84     amplify = amplify.t();
85     imshow("window",amplify);
86     imwrite("result.jpg",amplify);
87     waitKey(0);

```





## 5 对象移除

首先我们调用 opencv 自带的函数来选出要移除的选取，记录在 maskremove 里面。然后将图中这些要 remove 的东西的 energy function 中的权值置为一个很大的负权值，这样就可以在寻找 seam 的时候每次都删除掉 remove 区域里的点了。至于选择删除行还是列，这是使用的是贪心，如果行的 seam 值更小，就删除行。对于行列都计算一遍即可。

```
1 // 2. 创建一个可交互的窗口
2 Image showImg = input.clone(); // 拷贝一张图用于显示（因为需要在显示的图上面高
   亮标注，从而造成修改）
3 cv::namedWindow("Draw ROI", CV_WINDOW_AUTOSIZE); // 新建一个窗口
4 vector<vector<int>> maskRemove(row, vector<int>(col, 0)); // 希望获取的待删
   除选区
5 MouseArgs *args = new MouseArgs(showImg, maskRemove, Color(0, 0, 255)); // 攒一
   个MouseArgs结构体用于交互
6 cv::setMouseCallback("Draw ROI", onMouse, (void*)args); // 给窗口设置回调函数
7
8 // 拖动鼠标作画
9 while (1)
10 {
```

```

11 cv::imshow("Draw ROI", args->img);
12 // 按 esc 键退出绘图模式，获得选区
13 if (cv::waitKey(100) == 27)
14     break;
15 }
16 // maskRemove[200][400] = 1;
17 bool** choosed = new bool*[row];
18 for (int i = 0; i < row; ++i)
19     choosed[i] = new bool[col];
20 for (int i = 0; i < row; ++i)
21     for (int j = 0; j < col; ++j)
22         choosed[i][j] = (maskRemove[i][j] == 1);
23 while(1)
24 {
25     row = temp.rows;
26     col = temp.cols;
27     bool finished = true;
28     for (int i = 0; i < row; ++i)
29         for (int j = 0; j < col; ++j)
30         {
31             // printf("233%d %d\n", i, j);
32             if (choosed[i][j]) finished = false;
33         }
34     if (finished) break;
35     T nowpic;
36     nowpic.pic = temp;
37     getInfo(nowpic, choosed);
38     if (nowpic.rseam < nowpic.cseam)
39     {
40         temp = nowpic.rowpic;
41         row = temp.rows;
42         col = temp.cols;
43         // printf("%d %d\n", row, col);
44         bool** gg = new bool*[row];
45         for (int i = 0; i < row; ++i)
46             gg[i] = new bool[col];
47         for (int i = col - 1; i; --i)
48             for (int j = 0, k = 0; j < row; ++j)
49                 if (nowpic.rowseam[col - 1 - i].y != j)
50                     gg[k++][i] = choosed[j][i];
51         for (int i = 0; i < row + 1; ++i)
52             delete[] choosed[i];
53         delete[] choosed;
54         choosed = gg;
55     }
56     else
57     {
58         temp = nowpic.colpic;
59         row = temp.rows;
60         col = temp.cols;
61         // printf("%d %d\n", row, col);
62         bool** gg = new bool*[row];
63         for (int i = 0; i < row; ++i)
64             gg[i] = new bool[col];
65         for (int i = row - 1; i; --i)
66             for (int j = 0, k = 0; j < col; ++j)
67                 if (nowpic.colseam[row - 1 - i].x != j)
68                     gg[i][k++] = choosed[i][j];
69         for (int i = 0; i < row; ++i)
70             delete[] choosed[i];
71         delete[] choosed;
72         choosed = gg;
73     }

```

```
74 }
75 // 4. 垃圾回收
76 cv::setMouseCallback("Draw ROI", NULL, NULL); // 取消回调函数
77 delete args; // 垃圾回收
78 // waitKey(0);
79
80 // imshow("window",temp);
81 imwrite("result.jpg",temp);
82 waitKey(0);
```

代码流程也就像我说的一样，先获取 remove 区域，然后对于 remove 区域中的点的 energy 设一个负权值，然后不停地删除 seam 直到删完 remove 区域中的点。

下面是效果：

